
**Curso Superior de Tecnologia em Análise e
Desenvolvimento de Sistemas
Prof. Edécio Fernando Iepsen**

Algoritmos e Estruturas de Dados I

RECUSIVIDADE



Recursividade

Recursividade é um mecanismo que permite que uma função ou sub-rotina se invoque a si mesma, até atingir um caso base.

Algo que se consegue repetir pela aplicação da mesma regra, várias vezes.

Exemplos: Fatorial, Contagem Regressiva, Sequência de Fibonacci.

Como funciona a recursão?

Toda função recursiva
precisa de **dois elementos**:



Caso Base
(condição de
parada)



Chamada Recursiva
(A função se chama
novamente com um
problema menor)

Exemplo simples

Contagem Regressiva

* A função continua chamando a si mesma com $n-1$ até $n == 0$.

```
def contagem_regressiva(n):  
    # Caso base  
    if n == 0:  
        print("FIM!")  
    else:  
        print(n)  
        # Chamada recursiva  
        contagem_regressiva(n - 1)
```

```
contagem_regressiva(5)
```

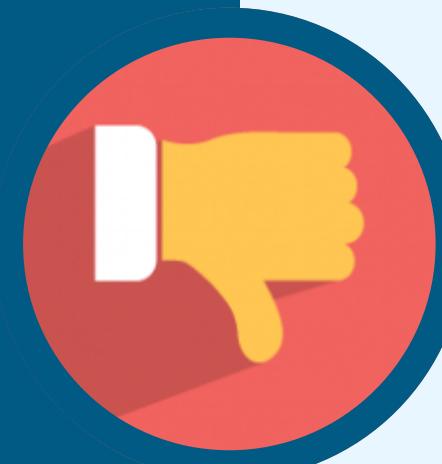
Benefícios e Desafios da Recursão



Vantagens:

Código mais elegante e legível para problemas recursivos.

Útil para dividir um problema grande em problemas menores.



Desvantagens:

Pode consumir muita memória se não for bem implementada.

Precisa de um caso base para evitar chamadas infinitas.

Exemplo Clássico

Fatorial

* O fatorial de 5! se expande como $5 * 4 * 3 * 2 * 1$.

```
def factorial(n):  
    # Caso base  
    if n == 1:  
        return 1  
    # Chamada recursiva  
    return n * factorial(n - 1)  
  
print(factorial(5))
```

Pilha de Execução (Call Stack)

Chamadas recursivas criam uma pilha de execução. Quando o caso base é atingido, as chamadas anteriores vão sendo desempilhadas e os valores são multiplicados.



`fatorial(5) → fatorial(4) → fatorial(3) → fatorial(2) → fatorial(1)`

Recursão x Interação

Recursão

- Mais legível em certos casos
- Usa pilha de chamadas
- Pode estourar a pilha para inputs grandes

Interação

- Mais eficiente em uso de memória
- Usa loops (for/while)
- Melhor para grandes entradas

É você programador que define se quer criar uma função recursiva ou não.

Exercícios:

Crie um programa para ler uma palavra e exibia-a de forma invertida - usando recursividade. Resolva de 2 formas: seguindo o exemplo simples (slide 4) e o exemplo clássico (slide 6).

